**What is Animation?**

Generate perception of motion with sequence of images shown in rapid succession
- humans “see” smooth motion at 12–70 fps

Must be technically excellent, but more importantly, aesthetically, emotionally compelling
- violation of realism may at times be desirable

**Animation “pipeline”:**

![Image of animation pipeline]

**Traditional Animation**

1. **Straight ahead:** draw each frame, one frame at a time
   - lead to spontaneity
   - great control
   - tedious: 24 fps, 1,440 frames/minute, 130K frames for a 1.5 hour movie

2. **Pose-to-pose (developed by Walt Disney):**
   - director plans shots using storyboards
   - senior artists sketch key poses (keyframes)
   - typically when motion changes
   - interns fill in the in-between frames
   - all line drawings are painted on cels
   - composed in layers
   - background changes infrequently, can be reused
   - photograph finished cel-stack onto film

**Computer Animation**

2D animation:
- CADrawing and painting are now routine
- but 2D in-betweening (morphing) is hard to get right

Instead, we assume 3D model of scene
- for each scene, vary parameters to generate desired pose for all objects
- **stop-motion:** shooting miniature physical models frame by frame
Some Artistic Considerations

Goal: make characters that move in a convincing way to communicate personality and emotion

Animation principles developed by Disney in the 20’s–30’s, adapted by CG animators, e.g., Lasseter of Pixar (now Disney)

The most important source of traditional animation principles is the book by Thomas and Johnston, *Disney Animation: The Illusion of Life*

Principles of Traditional Animation

Eleven principles of traditional animation compiled by Lasseter:

1. Squash and stretch
2. Slow in, slow out
3. Timing
4. Anticipation
5. Follow through
6. Overlapping action
7. Secondary action
8. Arcs
9. Exaggeration
10. Appeal
11. Staging

Many of these principles follow indirectly from physics, e.g., anticipation, follow-through, and many other effects can be produced by simply minimizing physical energy, but exaggerated

Squash and Stretch and Slow In/Out

**Squash and stretch**
- rigidity/softness via distortion during motion
- pseudo-physics: carrying momentum
- increase the sense of speed
- try to keep the volume constant

**Slow in and out**
- an extreme pose can be emphasized by slowing down as you get to it (and as you leave it)
- in practice, many things do not move abruptly but start and stop gradually
- pseudo-physics: overcoming inertia

Timing of Motion

Timing can completely change the interpretation of motion

Time spent on action affects perception
- timing indicates *weight*
- speed determines *emotion*

Since timing is so critical, animators usually draw a time scale next to keyframes to indicate how to generate the in-between frames
Anticipation

An action can be divided into three:
• anticipation
• action
• reaction

Anatomical motivation: a muscle must extend before it can contract

Prepares audience for an action
• don’t surprise the audience
• direct their attention to what’s important

Amount of anticipation can affect perception of speed and weight

Secondary Actions and Arcs

Use secondary actions to increase complexity of scene, but it should not interfere with the primary action

Avoid straight lines since most things in nature move in arcs

Exaggeration and Appeal

Exaggeration
• get to the heart of the idea and emphasize it so the audience can see it
• choose which properties to exaggerate

Appeal
• the character must interest the audience
• it doesn’t have to be cute and cuddly
• design, simplicity, behavior all affect appeal
• avoid perfect symmetries
• example: Luxo, Jr.
  was made to appear childlike

Follow Through and Overlapping Action

Actions do not end abruptly
• hand continues to move after throwing a ball: inertia
• audience likes to see resolution of action
• discontinuities are unsettling
• the termination of an action anticipates the next
• overlaps indicate intentions
Staging

Present the idea so that it is unmistakably clear
- audience can only focus on one thing at a time:
  main object should be **contrasted**
- stage action in silhouette
- in dialogue, characters should face \( \frac{3}{4} \) towards the camera, not right at each other

What is Animation?

**Make objects change over time**

Key technical problems are how to specify, generate, and manipulate **motion**

Four alternatives:

1. brute force: model each frame
2. key-frame animation:
   - key poses specified by hand
   - or poses recorded by **motion capture**
   - interpolate in-between frames

3. procedural/behavioral:
   - describe motion algorithmically
   - local rules, global **emergent behavior**: boids, brain-spring
   - **procedural texture**: crack propagation in glass or concrete, metallic patina, stone aging, water flow and rust

4. physical simulation:
   - motion according to physical laws
   - **particle systems** for fire, smoke
   - mass-spring damper arrays for fluttering cloth
   - fluid simulation

Key-frame Character Animation

Two approaches:
1. blend shapes or morph targets
2. rigged characters or articulated models

Blend shapes:
- a very simple surface control scheme
- no skeleton
- based on interpolating (tweening) among several key poses
- given a number of base meshes, combine them with time-dependent coefficients
Blend Shapes

Setup:
• user provides key shapes with a position \( \mathbf{p}_{ij} \) for every control point \( i \) in shape \( j \)
• for each frame \( k \) user provides a weight \( w_{j,k} \) for each key shape \( \sum_j w_{j,k} = 1 \), i.e., how much each key shape affects frame \( k \)
• the shape for frame \( k \) can be computed from the control points:
\[
\mathbf{p}_k = \sum_j w_{j,k} \mathbf{p}_{ij}
\]

Works well for relatively small motions
• runs in real time, e.g., as vertex shader
• often used for facial animation
• popular for games

Rigged Character

To support more complex character animation, models are often based on jointed skeletons (articulated models)

Kinematics: the study of movement of articulated models
• began in the mechanical engineering of robots

The skeleton can be fleshed out in any way, e.g., with mesh skinning
• skin deforms following bone movements

Articulated Model

Setup:
• rigid parts: each link (bone) in the articulated chain (skeleton) is rigid
• not physically accurate, but rigid skeleton constrains movement
• connected by joints: movement is constrained by the degree of freedom at each joint

Can be animated by specifying the joint angles as functions of time

Degrees of Freedom (DoFs)

Hinge/pin joint: 1 DoF
• knee or elbow joint

Saddle: 2 DoFs
• wrist/hand joints

Ball/socket joint: 3 DoFs
• hip, shoulder, neck
**Key-frame Animation**

With the character rigged, next specify key poses at specific time steps:

- Each pose controlled by a set of variables: joint angles, positions, etc.
- Each variable changes as a function of time.
- For each variable, specify its key value at “important” or key frames.
- In-between frames will be created by interpolating these key values.
- More generally, each variable may have a different set of key frames.

**Steps:**

1. **Character modeling:** design the geometry.
2. **Character rigging:** set up a bunch of parameters.
   - Joint angles, positions, etc.
3. **Set up key-framing:**
   - Specify key values of parameters at specific times.
   - By forward kinematics.
   - By inverse kinematics.
   - Incl. motion capture.
   - Specify an interpolation for the in-between values.

**Forward Kinematics**

Calculate the position and orientation of a limb’s end point (end effector) as a function of the angles of all joints:

\[ \mathbf{p} = f(\Theta) \]

Where:
- \( \mathbf{p} \): position of end effector
- \( \Theta \): angles, positions, ... of joints

For example:

\[
\begin{bmatrix}
    p_x \\
    p_y
\end{bmatrix} =
\begin{bmatrix}
    g(\phi, \varphi, \theta) \\
    h(\phi, \varphi, \theta)
\end{bmatrix}
\]

**Hierarchical Modeling**

Could animate by moving every control point (joint) at every key frame: tedious, hard to get smooth, consistent motion.

Animation need to be controlled at a higher level:
- “bend elbow” instead of
- “move left forearm one square inch”

Model objects as a hierarchy of components:
- Encodes topology (what’s connected to what).
- Specifies geometric relations from joints:
  - Tree structure
  - Each component defined relative to parent
  - Independent of display geometry
Moving Components

Define a coordinate system at the top of the hierarchy and at each joint.

Each joint is thus a separate frame of reference, with its own local coordinate system.

Each local coordinate system is defined in the coordinate system of the previous frame.

We can express positions and directions for frame \( i \) in the coordinate system of frame \( i-1 \) by a matrix transformation.

Embedding Transformations

To draw an object, traverse its hierarchical model.

Two types of nodes:
- object nodes: draw them (may include attributes)
- transform nodes:
  - multiply into Current Matrix on the way down
  - remove from Current Matrix on the way up

Constantly changing local coordinate system allows changes at different scales.

- apply rotation above “Upper Body” vs. rotation above “L Arm”

Be careful about transformation order:
- (usually) scale before rotate
- (usually) rotate before translate

Use OpenGL Matrix Stack

```
human()
{
    pelvis();
    glPushMatrix();
    glTranslate(...);
    glRotate(...);
    lhip();
    ...
    glPopMatrix();
    glPushMatrix();
    ...
    glPopMatrix();
    ...
}
```

Example

Out of two bones:

We want to model an arm as a hierarchy:

where:
- \( U \): upper arm
- \( L \): lower arm (forearm)

User controlled parameters:
- \( \phi \): shoulder joint angle
- \( \theta \): elbow joint angle
- \( t \): where shoulder meets torso
- \( w \): wrist joint location
**Positioning the Forearm**

Initially, segments in same position
- \( s \): shoulder joint location
- \( e_1, e_2 \): elbow joint locations
- \( w \): wrist joint location

First, perform elbow rotation
- translate elbow joint to origin
- rotate by given angle (\( \theta \))

(1) translate\((-e_2)\)
(2) rotate(\(\theta\))

**Attaching Forearm to Upper Arm**

Second, align corresponding elbow

(3) translate\((e_1)\)

Third, perform shoulder rotation
- operate on **whole** arm

(4) translate\((-s)\)
(5) rotate(\(\phi\))

**Placing the Shoulder**

Fourth, put shoulder in place

(6) translate\((t)\)

Important things to notice
- limited control knobs (just the angles)
- automatically handle interconnection (elbow joint)

**Converting to Hierarchy**

(1) translate\((-e_2)\)
(2) rotate(\(\theta\))
(3) translate\((e_1)\)
(4) translate\((-s)\)
(5) rotate(\(\phi\))
(6) translate\((t)\)
### Properties of Hierarchy

Geometry is always at the leaves
- Internal nodes are transform nodes

There are two types of transforms
- **Structural**
  - Fixed at design time
  - Keeps things together
- **Control knobs**
  - Variable parameters
  - Controlled by user

### Inverse Kinematics

**Forward kinematics**: the angles of all joints are explicitly specified by the animator
\[ x = L_1 \cos \theta_1 + L_2 \cos(\theta_1 + \theta_2) \]
\[ y = L_1 \sin \theta_1 + L_2 \sin(\theta_1 + \theta_2) \]

This gets tedious ... even with hierarchical modeling

**Inverse kinematics**: the animator drags the hands and feet into place and inverse kinematics solves for the angles to achieve the final position.

### Inverse Kinematics Example

Determine joint angles from position of end effector

First compute \( \theta_2 \): by cosine rule,
\[ |C|^2 = |A|^2 + |B|^2 - 2|A||B| \cos \theta \]
\[ x^2 + y^2 = L_1^2 + L_2^2 - 2L_1L_2 \cos(180 - \theta_2) \]
\[ \theta_2 = \cos^{-1}\left( \frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1L_2} \right) \]

Then solve for \( \theta_1 \) by expanding \( x \) and \( y \) (from prev slide)
using:
\[
\begin{align*}
\cos(\varphi + \psi) &= \cos \varphi \cos \psi - \sin \varphi \sin \psi \\
\sin(\varphi + \psi) &= \sin \varphi \cos \psi + \sin \psi \cos \varphi
\end{align*}
\]
\[ \theta_1 = \tan^{-1}\left( \frac{-(L_2 \sin \theta_2)x + (L_1 + L_2 \cos \theta_2)y}{(L_2 \sin \theta_2)y + (L_1 + L_2 \cos \theta_2)x} \right) \]

Unfortunately, real models are much more complex:
- A human has around 200 degrees of freedom

Suppose we specify locations of end effectors
- The mapping of parameters to effector positions is non-linear
  - Inverting this function is not possible
- We need to calculate the relative positions of all intermediate links to achieve the pose
  - This is an ill-posed problem (there may be infinitely many solutions for some chains)
    - Need to find a constrained solution, minimizing for example, the joint movements, maintain balance, ...
  - Similarly, there may not be any parameter settings that work
    - Need to pick one that is “close enough”
- Both involve some kind of optimization algorithm that rely on numerical methods, e.g., the Jacobian
Motion Capture

Instead of specifying end effector positions manually, measure it from the real world
Captures style, subtle nuances and realism

Motion Capture Technologies

Mechanical
- measure joint angles directly
- works in any environment
- restricts motion

Optical Passive
- strap a bunch of passive markers on subject (body, face)
- location of markers tracked by 8 or more cameras
- triangulate to get marker’s 3D position
- convert this to joint angles and map to articulated model
- high frequency (240 Hz)
- restricted volume: studio size, lighting, number of cameras
- occlusions are troublesome

Magnetic
- tethered or wireless
- transmitter emits field
- trackers sense field
- trackers report position and orientation

Disadvantages:
- nearby metal objects cause distortions
- limited range
- limited number of trackers
- low frequency (60 Hz)
Optical Motion Capture Process

1. Start with standard rest pose
2. Calibrate: match skeleton, find offsets to markers
3. Identify and uniquely label markers
4. Motion trial: use a short sequence that exercises all DOFs of the subject
5. Track forward through time (but watch for markers dropping out due to occlusion!)
6. Compute joint angles: explain data using skeleton DOFs
   ⇒ an inverse kinematics problem per frame!

Marker Data to Motion

Motion capture gives inconvenient raw data
• passive optical gives “least information”
• accurate position, but correspondence difficult
  • which marker is which?
  • where are the markers relative to the skeleton?

Motion Capture Technologies

Active Optical
• uses LEDs instead of passive markers
• LEDs blink IDs → correspondence automatic
• number of markers trades off with frame rate

Pros and Cons of Motion Capture

Mocap data is very realistic
• timing matches performance exactly
• dimensions are exact

But it is not enough for good character animation
• noise, errors from non-rigid marker mounting
• contains no exaggeration
• limited in the complexity of the scenes they can capture
Pros and Cons of Motion Capture

To increase versatility of mocap:
- break scenes into smaller pieces and re-construct later
- gather lots of snippets of motion capture
  - e.g.: several ways to dunk, dribble, pass
- arrange them so that they can be pieced together smoothly
- at run time, figure out which pieces to play for desired motion

Automated stop motion?

Problem: once the data is captured, it’s hard to modify for a different purpose

Performance Capture

Mocap is no panacea for assigning key values to parameters ⇒ mocap data is generally a starting point for skilled animators to create the final product

Many studios regard motion capture as low quality, cheap motion
- no directive/creative control

Performance capture is different
- use mocap device as an expressive input device
  - e.g., James Cameron’s Avatar

Chenney  O’Brien